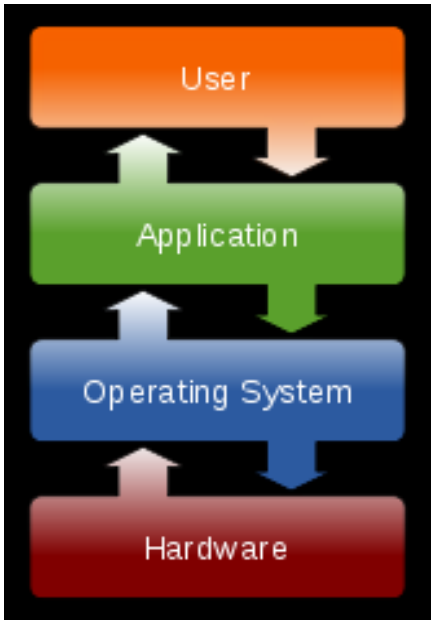
Sissejuhatus infotehnoloogiasse

Operatsioonisüsteemid



Miks opsüsteem?

Opsüsteemi põhieesmärgid:

- Pakkuda programmeerijale valmistehtud standardtükke.
- Võimaldada kasutajal arvutis ühtemoodi ja harjumuspäraselt tegutseda, sõltumatult sellest, mis programmid tal arvutis on.

Miks opsüsteem?

Arvutit saaks programmeerida ka ilma opsüsteemita. Sel juhul:

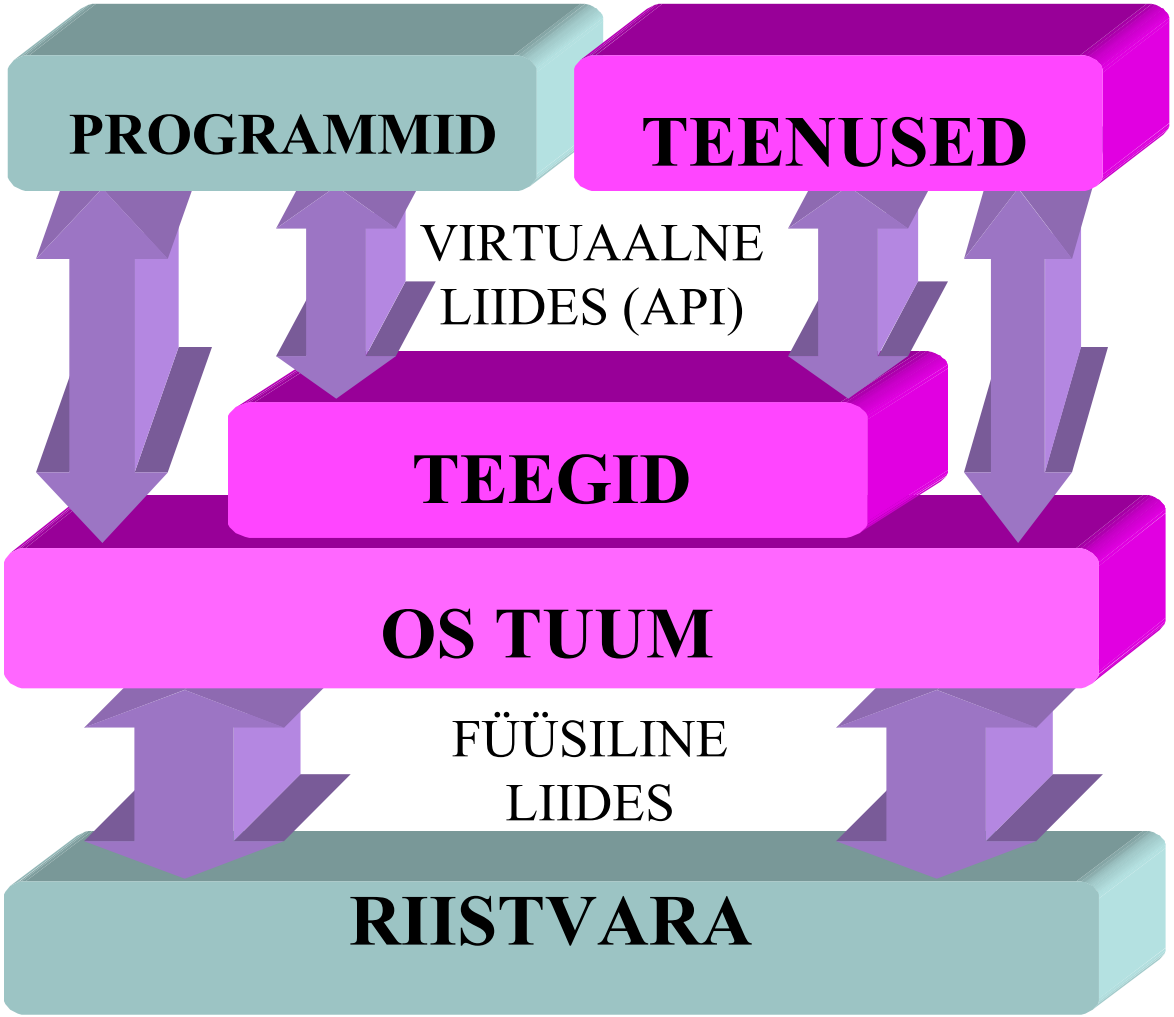
- oleks iga programmi tegemine palju raskem kui opsüsteemi olemasolu korral.
- kasutajate jaoks näeks eri programmid väga eri moodi välja.

Mida opsüsteem enamasti teeb?

- Oskab kettalt programme lugeda ja neid käima panna.
- Oskab programme seisma panna (lõplikult või ainult väikese pausi jaoks)
- Oskab kettale faile ja katalooge kirjutada ja sealt neid lugeda.
- Oskab välisseadmetega (printer, monitor, klaviatuur jne jne) suhelda.
- Oskab võrguga suhelda.
- **jne**

- **Kui opsüsteemi ei oleks, peaks iga programm kõiki neid asju ise teha oskama!**

Operatsioonisüsteemi roll



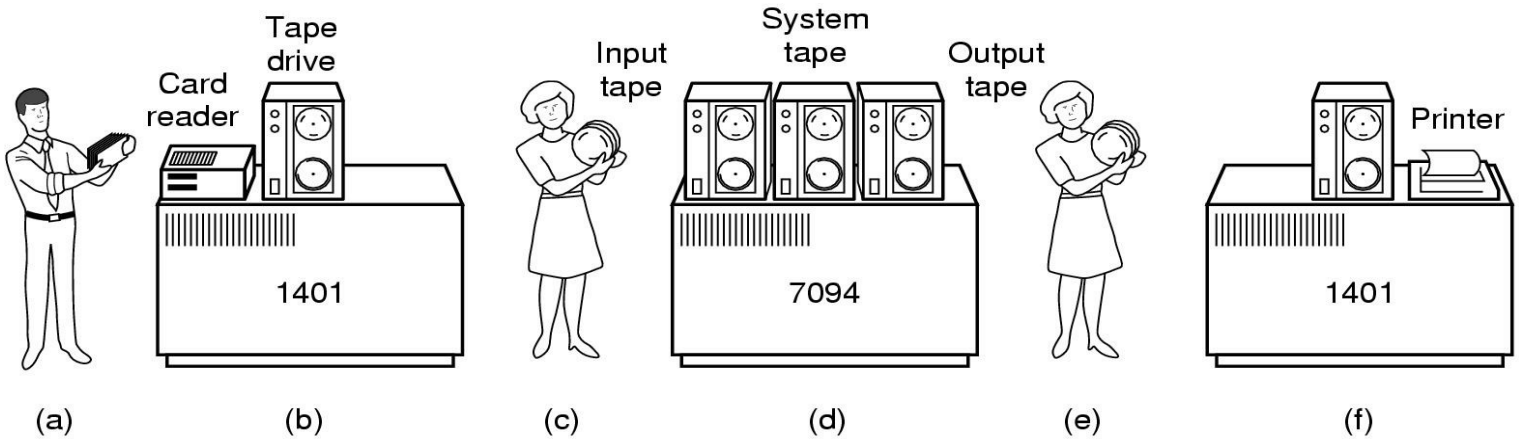
What *is* an operating system?

- **A program that runs on the “raw” hardware and supports**
  - Resource Abstraction
  - Resource Sharing
- **Abstracts and standardizes the interface to the user across different types of hardware**
  - Virtual machine hides the messy details which must be performed
- **Manages the hardware resources**
  - Each program gets time with the resource
  - Each program gets space on the resource
- **May have potentially conflicting goals:**
  - Use hardware efficiently
  - Give maximum performance to each user

Operating system timeline

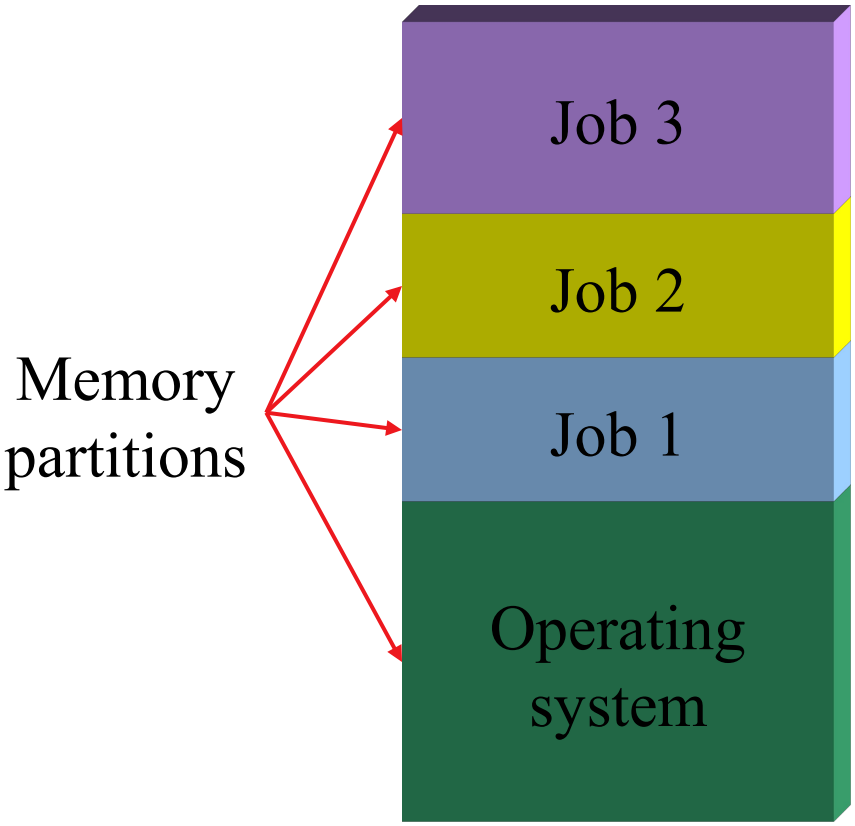
- **First generation: 1945 – 1955**
  - Vacuum tubes
  - Plug boards
- **Second generation: 1955 – 1965**
  - Transistors
  - Batch systems
- **Third generation: 1965 – 1980**
  - Integrated circuits
  - Multiprogramming
- **Fourth generation: 1980 – present**
  - Large scale integration
  - Personal computers
- **Next generation: ???**
  - Systems connected by high-speed networks?
  - Wide area resource management?

Second generation: batch systems



- Bring cards to 1401
- Read cards onto input tape
- Put input tape on 7094
- Perform the computation, writing results to output tape
- Put output tape on 1401, which prints output

Third generation: multiprogramming

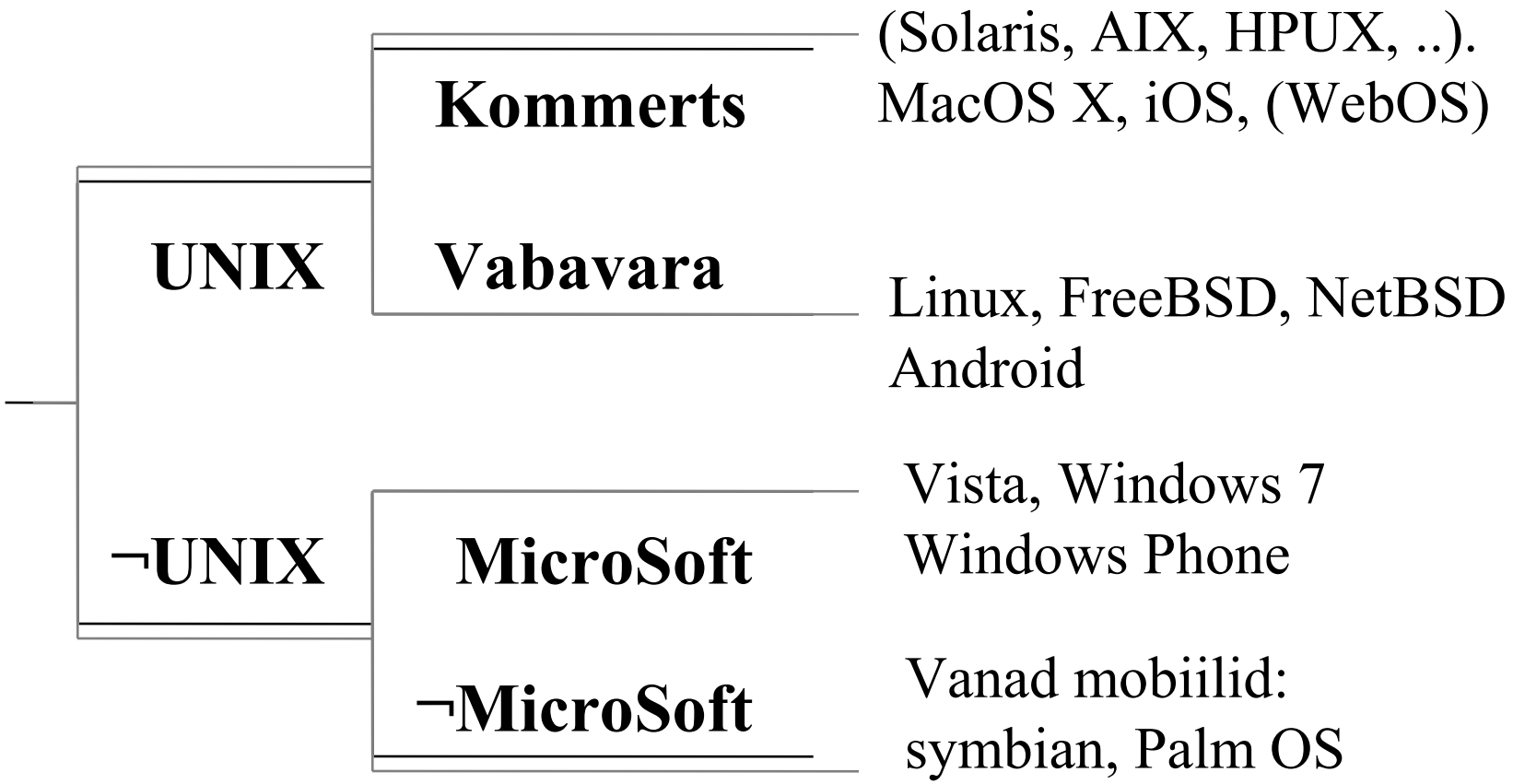


- Multiple jobs in memory
  - Protected from one another
- Operating system protected from each job as well
- Resources (time, hardware) split between jobs
- Still not interactive
  - User submits job
  - Computer runs it
  - User gets results minutes (hours, days) later

Natuke ajalugu (4. põlvkond)

1968	ARPANET
1969	AT&T Bell Labs UNIX
1977	BSD UNIX
1979	AT&T UNIX kommertskasutusse
1981	MS-DOS
1983	SCO UNIX
1984	SunOS
1987	OS/2
1987	MINIX (avatud koodiga õppe-opsüsteem)
1989	AT&T UNIX SVR4 (System V Release 4), POSIX
1991	Linux, WWW
1992	SUSE linux
1993	Windows NT
1993	Debian Linux
1994	RedHat Linux
2000	Windows 2000

OS-ide tüüpide puu



Main Microsoft OS-es: historical

- **MS-DOS** is a text-based desktop operating system made by Microsoft that runs on Intel 80x86.
- **Windows 98, Windows 95, and Windows 3.1** are desktop operating systems made by Microsoft that run on Intel Pentium and Intel 80x86.
- **Windows NT**, Windows NT Server, and Windows NT Server Enterprise Edition are server and workstation operating systems made by Microsoft that run on Intel Pentium, Intel 80x86, and DEC Alpha.
- **Windows 2000** Professional, Windows 2000 Server, and Windows 2000 Advanced Server are server and workstation operating systems made by Microsoft that run on the Intel Pentium.
- **Windows XP** is a server and workstation operating system made by Microsoft that run on the Intel Pentium. Converges 95 and NT/2000.
- **Windows Vista and 7** are windows XP evolutions.
- **Windows CE** is an old mobile version of Windows
- **Windows Phone** is a new mobile version of Windows

OS-es: Other IBM PC os-s: historical

- **PC-DOS-2000** is a text-based desktop operating system made by IBM as an update of the older MS-DOS. It runs on Intel 80x86. Only low end servers can run on this operating system.

- **OS/2** is a high performance desktop and high end operating system made by IBM that runs on Intel Pentium and Intel 80x86.

OS-es: Apple: historical

Macintosh OS 9, OS 8, OS 7, and OS 6 are desktop operating systems made by Apple Computers that ran on Motorola 680x0. and now on Motorola/IBM PowerPC

Darwin is a UNIX-based operating system that includes capabilities from BSD unix, the NeXT and Macintosh operating systems. Rhapsody was a foundation for OS X of the Macintosh.

Macintosh OS X (ten) is a desktop operating system based on Darwin. Macintosh OS X is made by Apple Computers and ran originally on Motorola/IBM PowerPC, now on Intel

iOS is a mobile operating system based on OS X. Runs on ARM family of processors

OS-es: main free UNIXes

All these UNIXes are actively developed:

- **LINUX** is a free version of UNIX that runs on Intel Pentium, Intel 80x86, Motorola/IBM PowerPC, Motorola 680x0, Sun SPARC, SGI MIPS, DEC Alpha, HP PA-RISC, DEC VAX, ARM etc
- **Android** is a free version/extension of Linux developed by Google and Open Handset Alliance
- **FreeBSD** is a free version of UNIX that runs on Intel Pentium and Intel 80486.
- **NetBSD** is a free version of UNIX that runs on Intel Pentium, Intel 80486, Intel 80386, Motorola/IBM PowerPC, Motorola 680x0, Sun SPARC, HP PA-RISC, DEC VAX, and ARM.
- **OpenBSD** is a free version of UNIX that runs on Intel Pentium, Intel 80486, Intel 80386, Motorola/IBM PowerPC, Motorola 680x0, Sun SPARC, HP PA-RISC, DEC VAX, and ARM.
- **GNU Hurd** is a free UNIX-based operating system.

OS-es: historical commercial UNIXes

All these commercial UNIXes except OS X, iOS and Solaris are dead by now:

- **Solaris** is a UNIX-based operating system made by Sun Computers that runs on Sun SPARC and Intel Pentium
- **OS X and iOS** are Apple operating systems based on Mach and various BSD unixes plus NextStep.
- **Sun-OS** is an older text-based UNIX that runs on Sun SPARC. Solaris is an enhancement of Sun-OS that includes a graphic user interface.

- **AIX** is IBM's version of UNIX.
- **HP-UX** is a UNIX-based operating system made by Hewlett-Packard that runs on HP PA RISC.
- **ULTRIX** is a UNIX-based operating system made by DEC. ULTRIX has been replaced by Digital UNIX.

OS-es: less common

- **IRIX** is a UNIX-based operating system made by SGI that runs on SGI MIPS.
- **NeXT** is a UNIX-based operating system made by NeXT that runs on Intel Pentium, Intel 80486, and Motorola 68040. OpenSTEP runs on Intel Pentium, Intel 80486, Motorola 68040, Sun SPARC, and HP PA RISC.
- **MVS** is a mainframe operating system made by IBM
- **VMS** is a high performance operating system made by DEC that runs on DEC VAX. OpenVMS is an updated version of VMS.
- **NetWare** "is a dedicated network operating system" that runs on Intel Pentium, Intel 80486, and Intel 80386.
- **BeOS** is a high performance desktop operating system made by Be that runs on Motorola/IBM PowerPC and Intel Pentium. Only low end servers can run on this operating system.
- **AmigaOS** is an old but popular operating system that is being resurrected. Only low end servers can run on this operating system.

UNIXi filosoofia

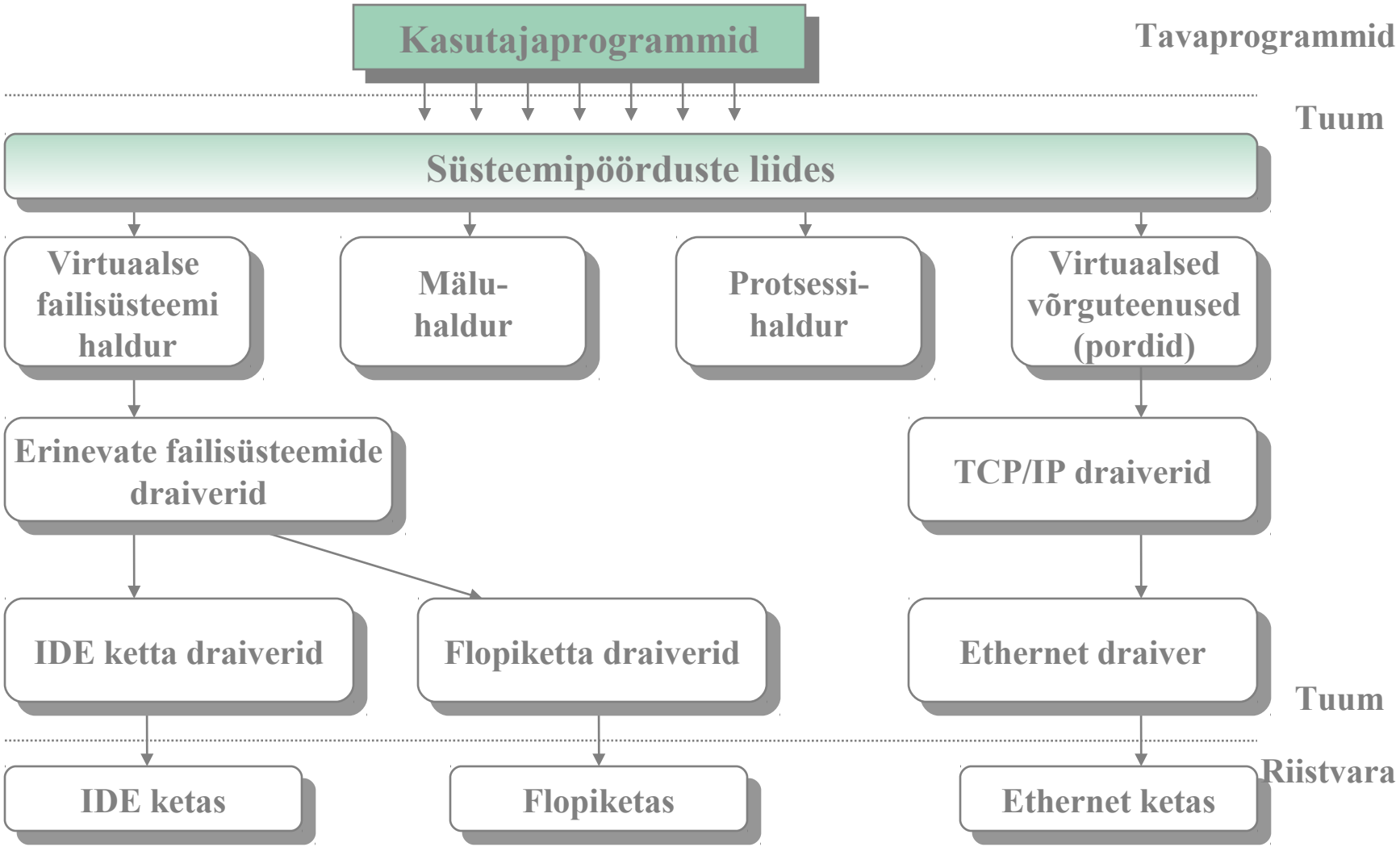
- **väike on ilus**
  - iga utiliit/teenus täidab ainult ühte funktsiooni
  - komponentide ehitus ja opsüsteemi vahendid võimaldavad kerget ja selge liidesega kombineerimist
  - keerulisemad teenused pannakse kokku lihtsamatest
  - tulemuseks on alternatiivsete lahenduste võimalus ja süsteemide mitmekihiline ehitus
- **kõik on fail**
  - failid, kataloogi, seadmed on ligipääsetavad sarnase liidesega
  - minimaalselt baasmõisteid, mis on võimalikult universaalsed

Failide paigutus

- **Windows**
  - Ketta nimed A:, B:, C: jne
  - programmid ja konfiguratsioon samades kataloogides

- **UNIX laadsed operatsioonisüsteemid**
  - järgivad ühtsed failisüsteemi paigutuse standardit
    FHS - www.pathname.com/fhs
  - failid on paigutatud laiali eri kataloogidesse otstarbe järgi
  - kasutavad eri failisüsteemide ühendamiseks monteerimist (*mounting*)

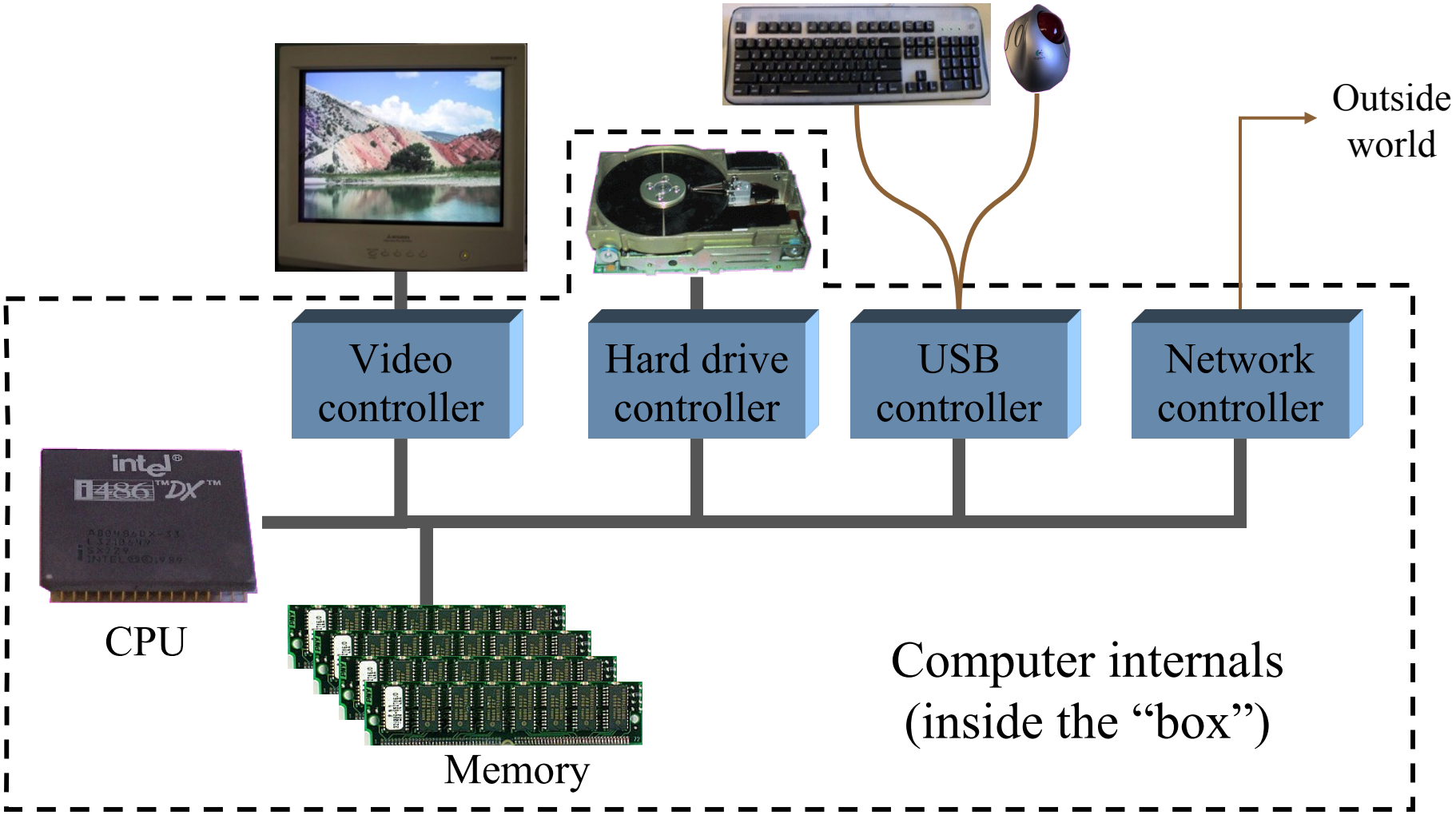
Operatsioonisüsteemi tuuma ehitus



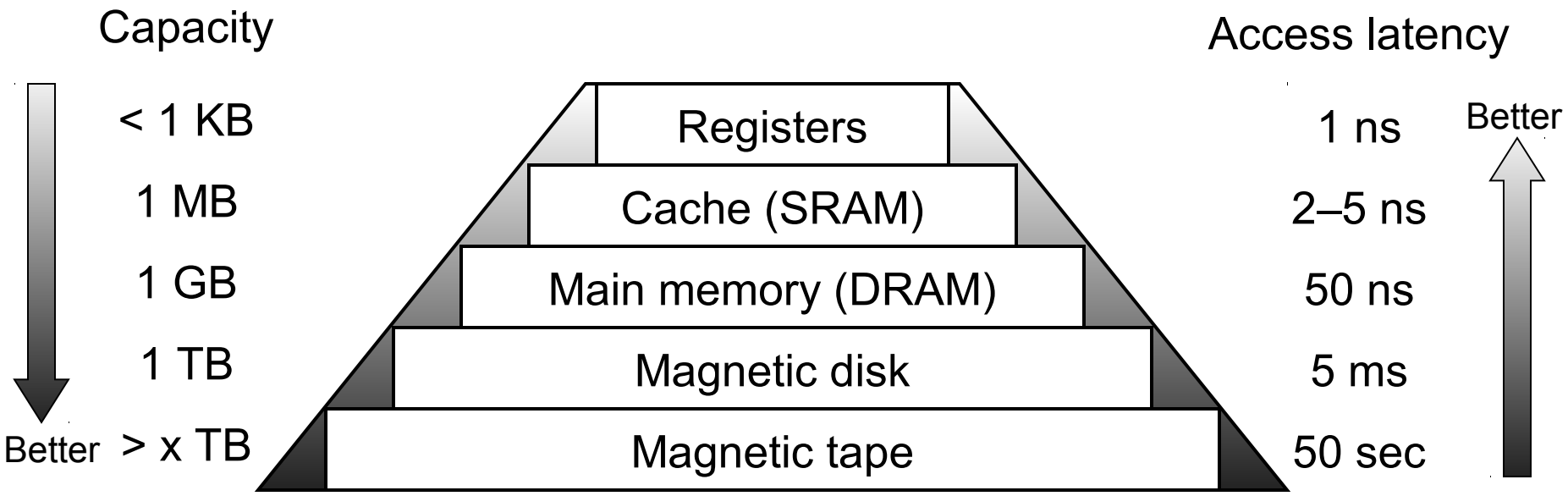
Tuuma ülesanded

- Loob protsessidele stabiilse ja üksteisest isoleeritud “elukeskkonna” ning kommunikatsioonivahendid
- Loob liidese failidele ja riistvararessurssidele
- Seostab kõik protsessid, failid jm omanikuga ja piirab protsesside pöördumise failide/ressursside poole vastavalt pääsu- ja kasutusõigustele

Components of a simple PC



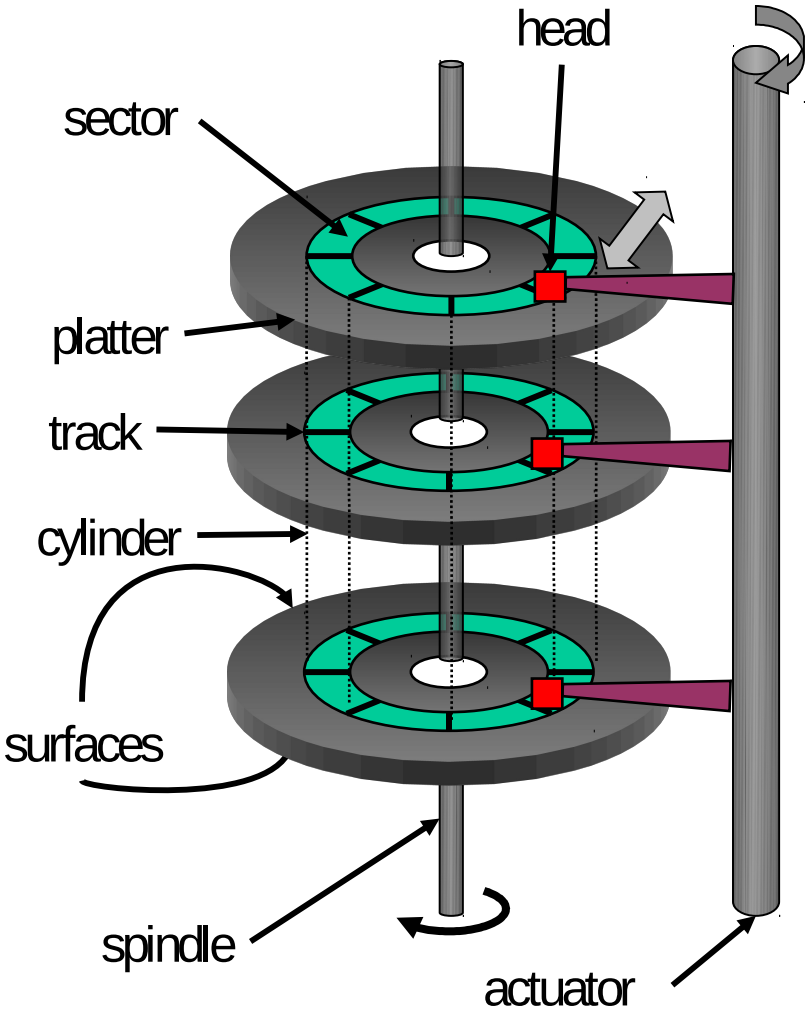
Storage pyramid



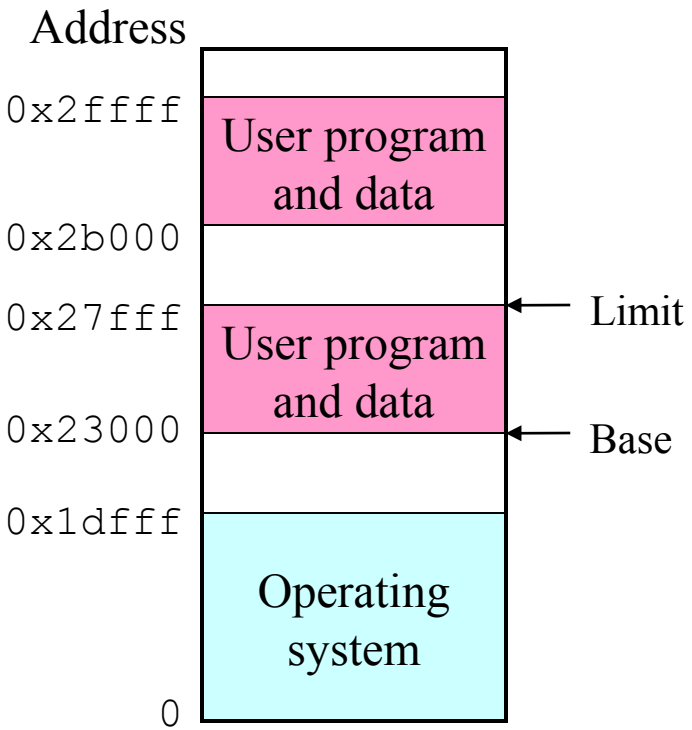
- Goal: really large memory with very low latency
  - Latencies are smaller at the top of the hierarchy
  - Capacities are larger at the bottom of the hierarchy
- Solution: move data between levels to create illusion of large memory with low latency

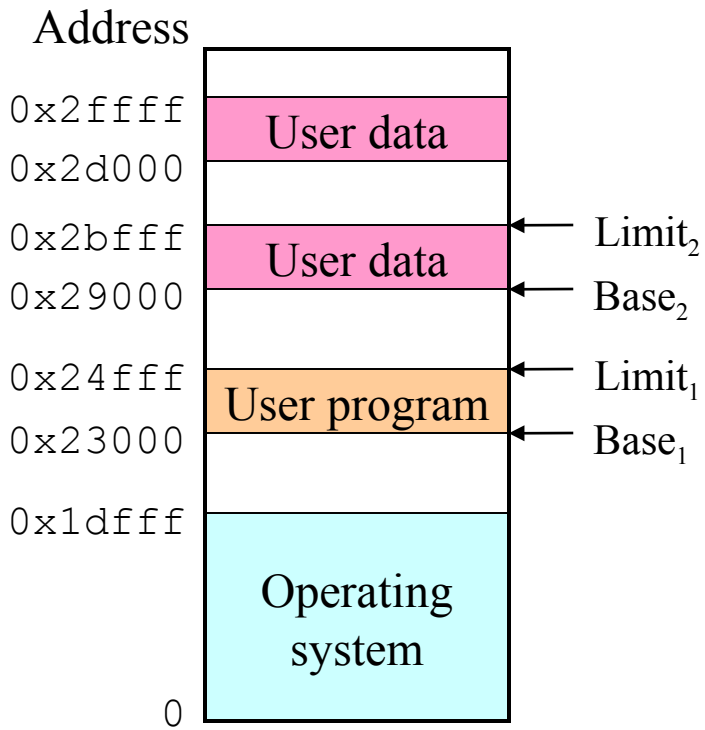
Disk drive structure

- Data stored on surfaces
  - Up to two surfaces per platter
  - One or more platters per disk
- Data in concentric tracks
  - Tracks broken into sectors
    - 256B-1KB per sector
  - Cylinder: corresponding tracks on all surfaces
- Data read and written by heads
  - Actuator moves heads
  - Heads move in unison



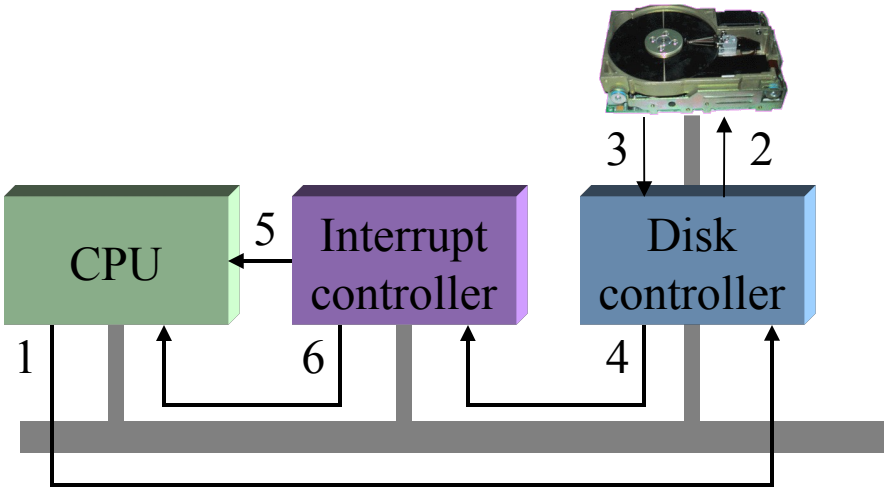
Memory

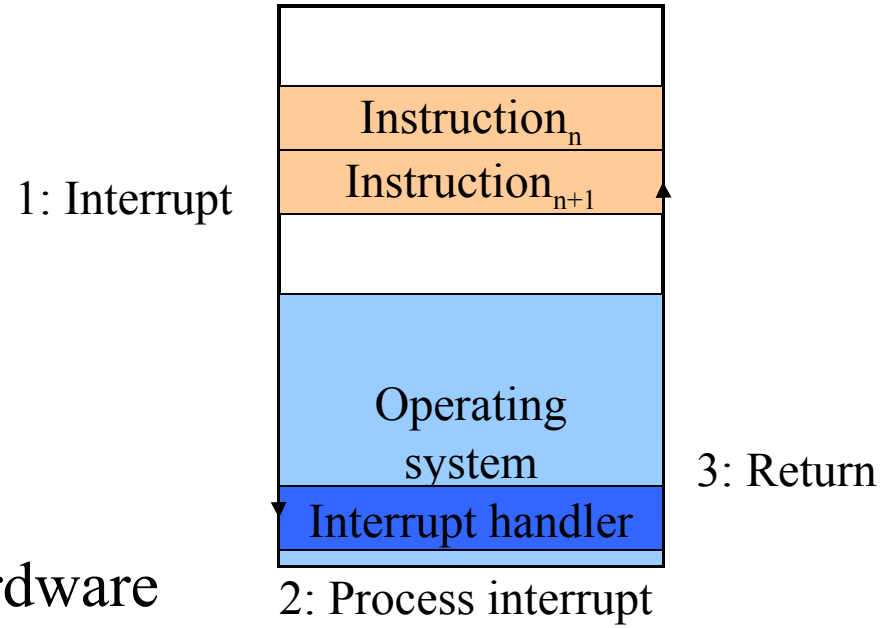




- Single base/limit pair: set for each process
- Two base/limit registers: one for program, one for data

Anatomy of a device request





- Left: sequence as seen by hardware
  - Request sent to controller, then to disk
  - Disk responds, signals disk controller which tells interrupt controller
  - Interrupt controller notifies CPU
- Right: interrupt handling (software point of view)

Interrupts

- The interrupt handler must save the machine state, do some processing, then call the process scheduler and dispatcher.

- When an interrupt occurs

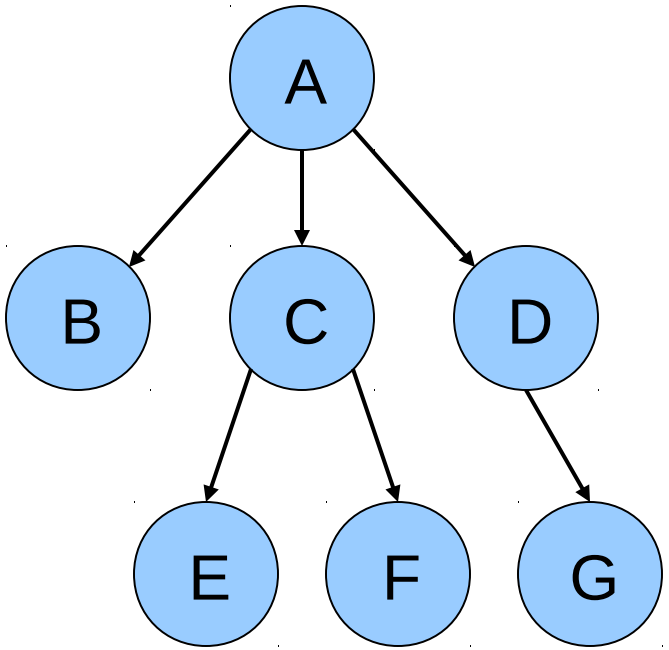
  1. the processor hardware makes a quick copy of the program counter and CPU registers
  2. the hardware switches to kernel mode and jumps to the interrupt service routine
  3. the ISR is usually very short. It may inform a device driver that it received the interrupt; it may just increment some clock counters.
  4. next the ISR calls the scheduler, which decides which process to run
  5. the scheduler calls the dispatcher, and new process (or maybe the same process) resumes where it left off

- An important goal of the OS is to hide interrupts from the user---and from user-level processes.

Operating systems concepts

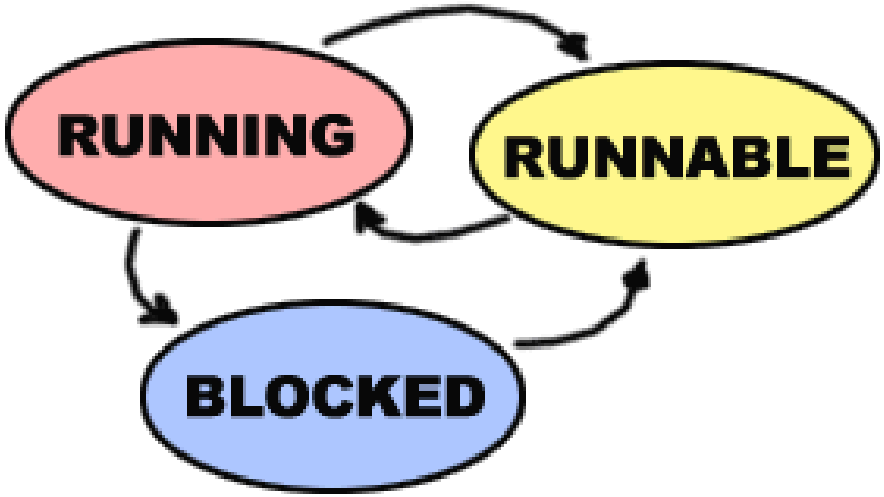
- Many of these should be familiar to Unix users...
- Processes (and trees of processes)
- Deadlock
- File systems & directory trees
- Pipes

Processes

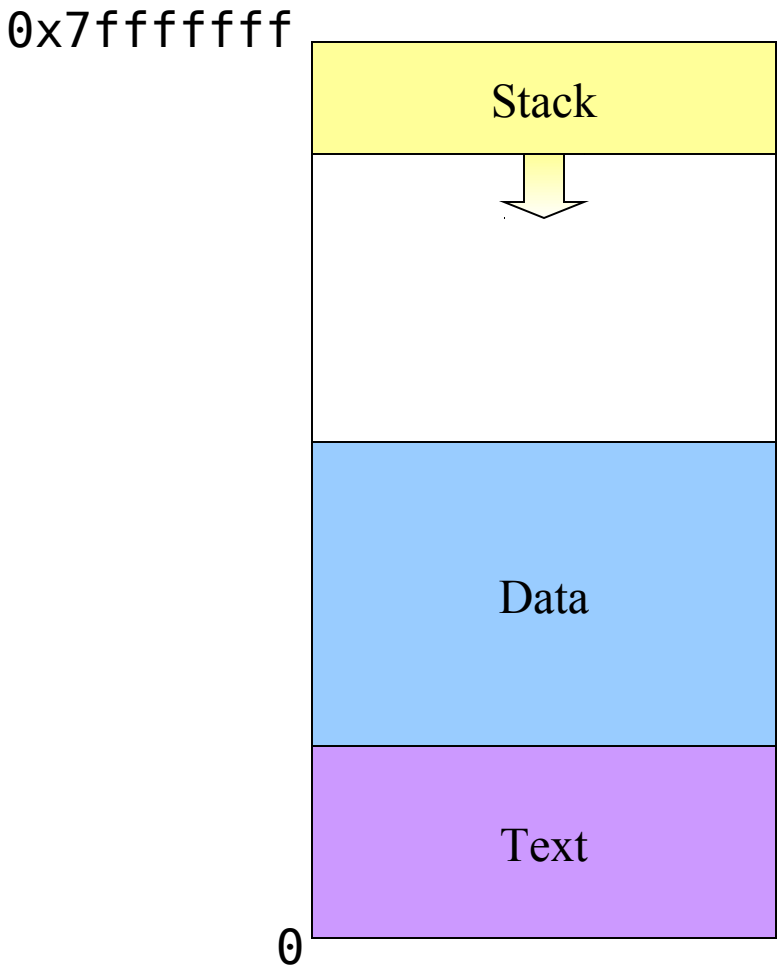


- Process: program in execution
  - Address space (memory) the program can use
  - State (registers, including program counter & stack pointer)
- OS keeps track of all processes in a *process table*
- Processes can create other processes
  - Process tree tracks these relationships
  - A is the *root* of the tree
  - A created three child processes: B, C, and D
  - C created two child processes: E and F
  - D created one child process: G

Processes wait and run



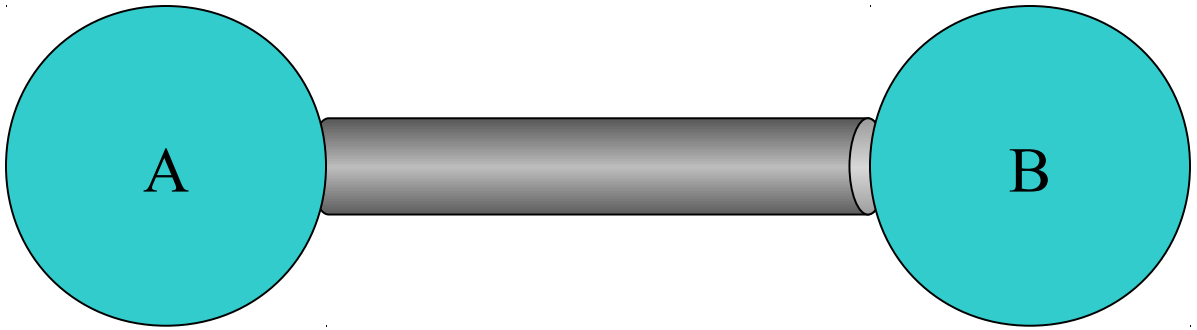
Inside a (Unix) process



- Processes have three segments
  - Text: program code
  - Data: program data
    - Statically declared variables
    - Areas allocated by `malloc()` or `new`
  - Stack
    - Automatic variables
    - Procedure call information
- Address space growth
  - Text: doesn't grow
  - Data: grows "up"
  - Stack: grows "down"

Interprocess communication

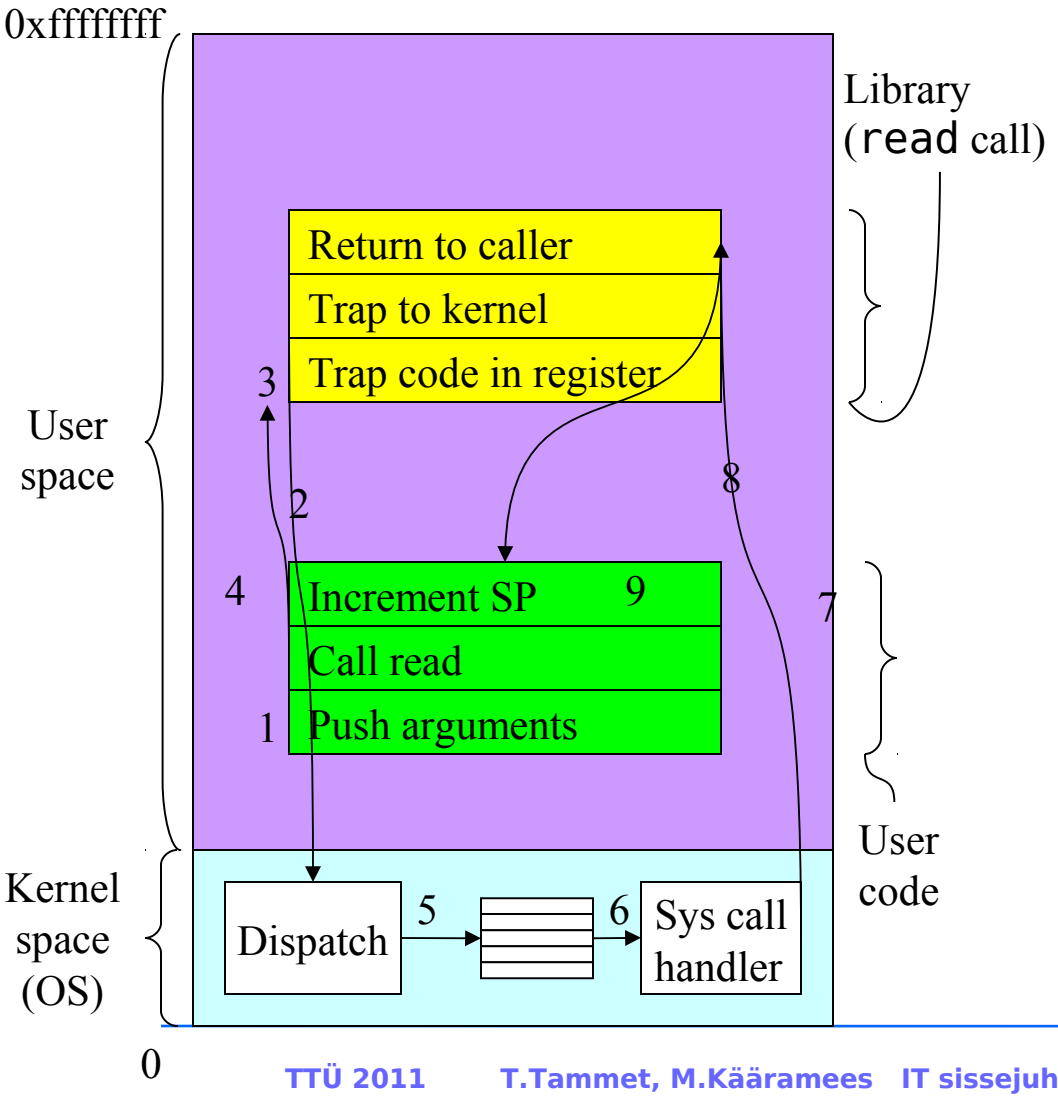
- Processes want to exchange information with each other
- Many ways to do this, including
  - Network
  - Pipe (special file): A writes into pipe, and B reads from it



System calls

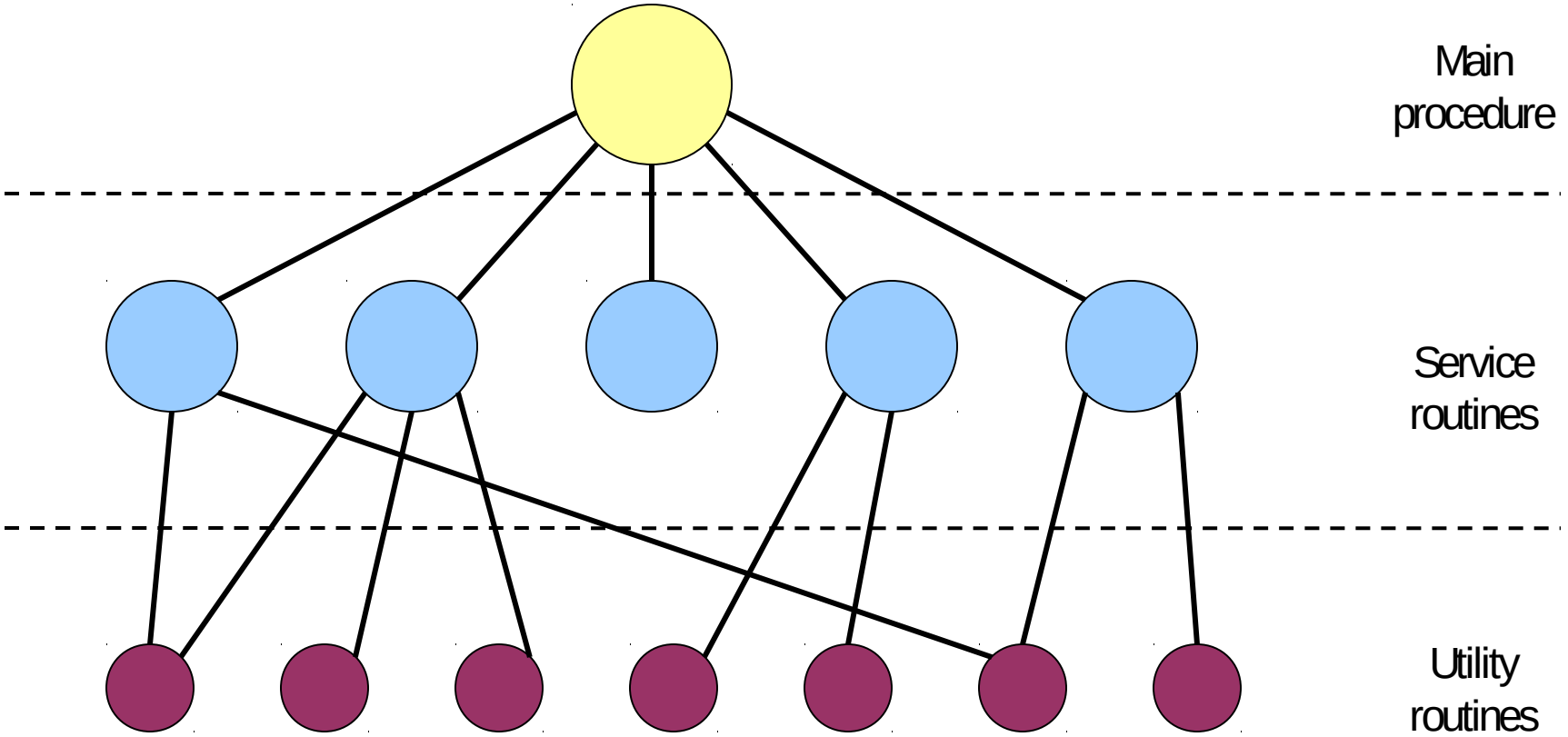
- **Programs want the OS to perform a service**
  - Access a file
  - Create a process
  - Others…
- **Accomplished by system call**
  - Program passes relevant information to OS
  - OS performs the service if
    - The OS is able to do so
    - The service is permitted for this program at this time
  - OS checks information passed to make sure it’s OK
    - Don’t want programs reading data into other programs’ memory!

Making a system call

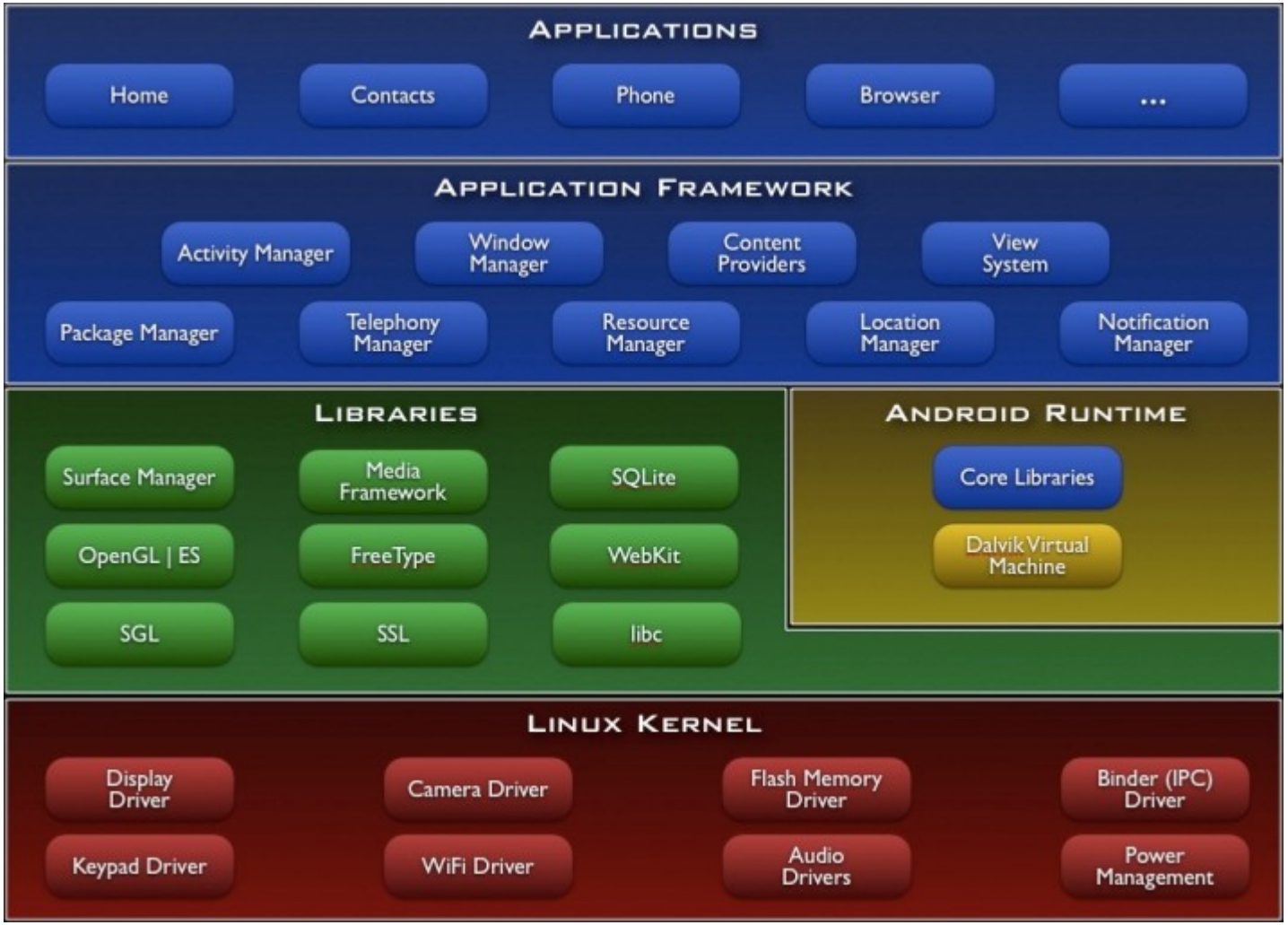


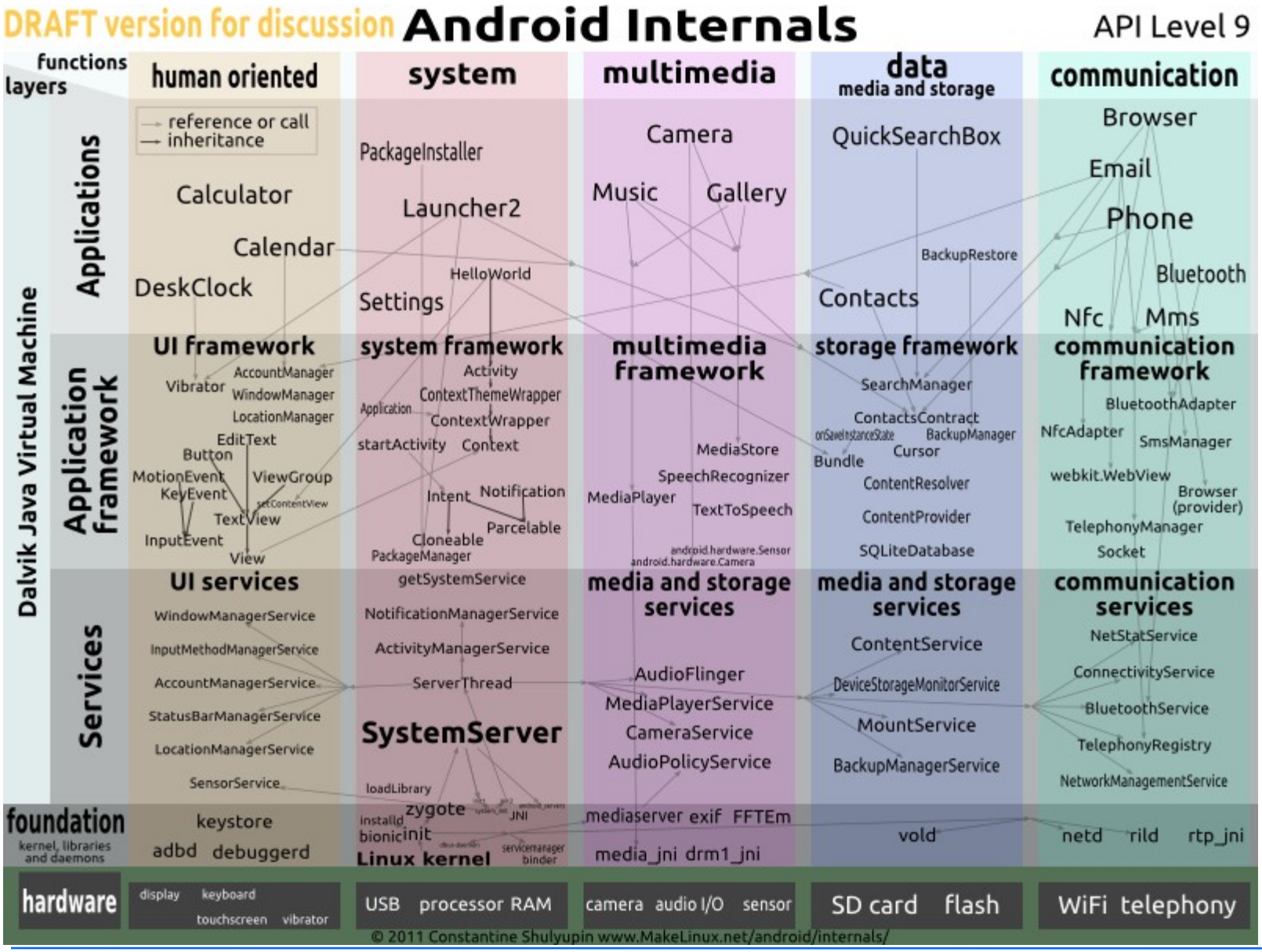
- System call: `read(fd,buffer,length)`
- Program pushes arguments, calls library
- Library sets up trap, calls OS
- OS handles system call
- Control returns to library
- Library returns to user program

Monolithic OS structure

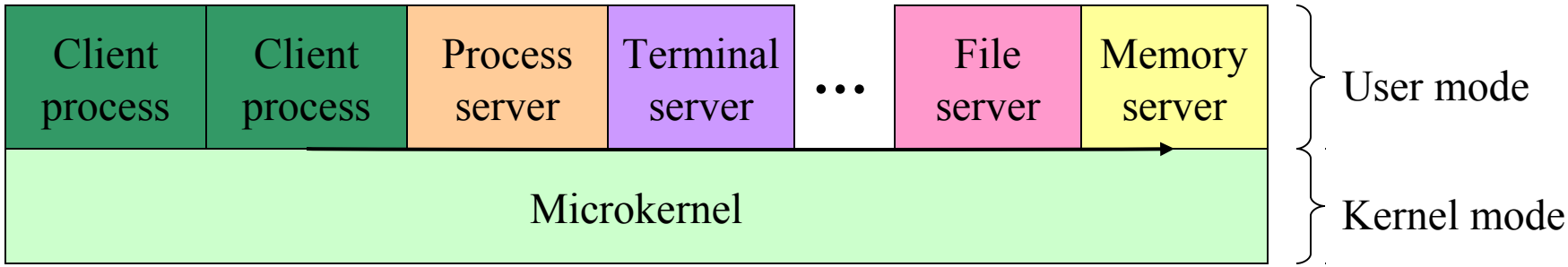


Android OS structure




DRAFT version for discussion
Android Internals
API Level 9
functions
layers
human oriented
system
multimedia
data
media and storage
communication
reference or call
inheritance
Applications
Calculator
Calendar
DeskClock
PackageInstaller
Launcher2
Settings
HelloWorld
Camera
Music
Gallery
QuickSearchBox
BackupRestore
Contacts
Browser
Email
Phone
Bluetooth
Nfc
Mms
Application framework
UI framework
Vibrator
AccountManager
WindowManager
LocationManager
EditText
Button
MotionEvent
KeyEvent
ViewGroup
setContentView
TextView
InputEvent
View
system framework
Activity
ContextThemeWrapper
Application
ContextWrapper
startActivity
Context
Intent
Notification
Parcelable
Cloneable
PackageManager
multimedia framework
MediaStore
SpeechRecognizer
MediaPlayer
TextToSpeech
android.hardware.Sensor
android.hardware.Camera
storage framework
SearchManager
ContactsContract
onSaveInstanceState
BackupManager
Bundle
Cursor
ContentResolver
ContentProvider
SQLiteDatabase
communication framework
BluetoothAdapter
NfcAdapter
SmsManager
webkit.WebView
Browser
(provider)
TelephonyManager
Socket
Services
UI services
WindowManagerService
InputMethodManagerService
AccountManagerService
StatusBarManagerService
LocationManagerService
SensorService
getSystemService
NotificationManagerService
ActivityManagerService
ServerThread
SystemServer
media and storage services
AudioFlinger
MediaPlayerService
CameraService
AudioPolicyService
media and storage services
ContentService
DeviceStorageMonitorService
MountService
BackupManagerService
communication services
NetStatService
ConnectivityService
BluetoothService
TelephonyRegistry
NetworkManagementService
Dalvik Java Virtual Machine
foundation
kernel, libraries and daemons
keystore
adbd
debuggerd
loadLibrary
zygote
JNI
installd
bionic
init
servicemanager
binder
Linux kernel
mediaserver
exif
FFTEm
media_jni
drm1_jni
vold
netd
rild
rtp_jni
hardware
display
keyboard
touchscreen
vibrator
USB
processor
RAM
camera
audio I/O
sensor
SD card
flash
WiFi
telephony
© 2011 Constantine Shulyupin www.MakeLinux.net/android/internals/

Microkernels (client-server)



- Processes (clients and OS servers) don't share memory
  - Communication via message-passing
  - Separation reduces risk of "byzantine" failures
- Main example: Mach

POSIX pääsuõigused

d	r w x	r w x	r w x
faili tüüp	omaniku õigused	grupi õigused	teiste õigused

r lugemisõigus
w kirjutamisõigus
x käivitamisõigus

s setUID, setGID
t sticky bit

Loabittide tähendused

Loabitt	Tähendus faile	Tähendus kataloogile
r	Faili saab lugeda	Saab kuvada kataloogi sisu (ls)
w	Faili saab redigeerida	Saab luua ja kustutada faile kataloogis
x	Faili saab käivitada (programm)	Saab kataloogi kasutada (sõlme leidmiseks)

SUID (s)	Programm käivitub omaniku õigustes	-
SGID (s)	Programm käivitub grupi õigustes	Kataloogi loodavad failid antakse kataloogi grupile
Sticky (t)	-	Faile saab muuta ja kustutada ainult omanik

Pääsuõigused NTFS failisüsteemis

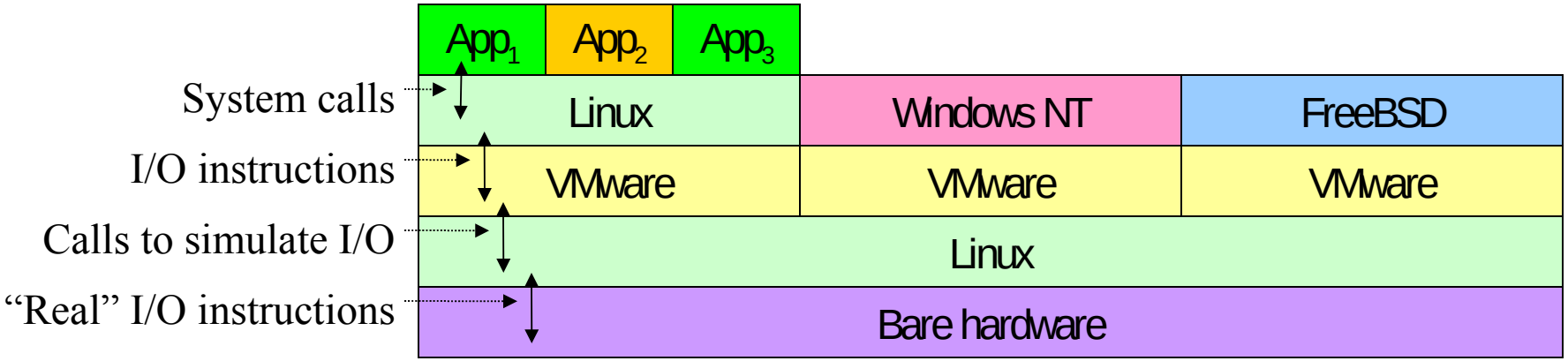
- **Iga faili ja kataloogiga on seotud pääsuõiguste nimekiri (ACL - Access Control List)**
- **ACL on nimekiri pääsuõiguste kirjetest (ACE - Access Control Entry)**
- **ACE koosneb:**
  - kasutaja, grupi või arvuti nimest
  - pääsuõiguste loetoelust
- **Kui kasutajale, ega tema grupile ei ole mingi ressursi juures vastavat õigust antud, siis pole tal võimalik seda ressurssi kasutada**

www.windowsitlibrary.com/Content/592/1.html

Failisüsteemide võrdlus

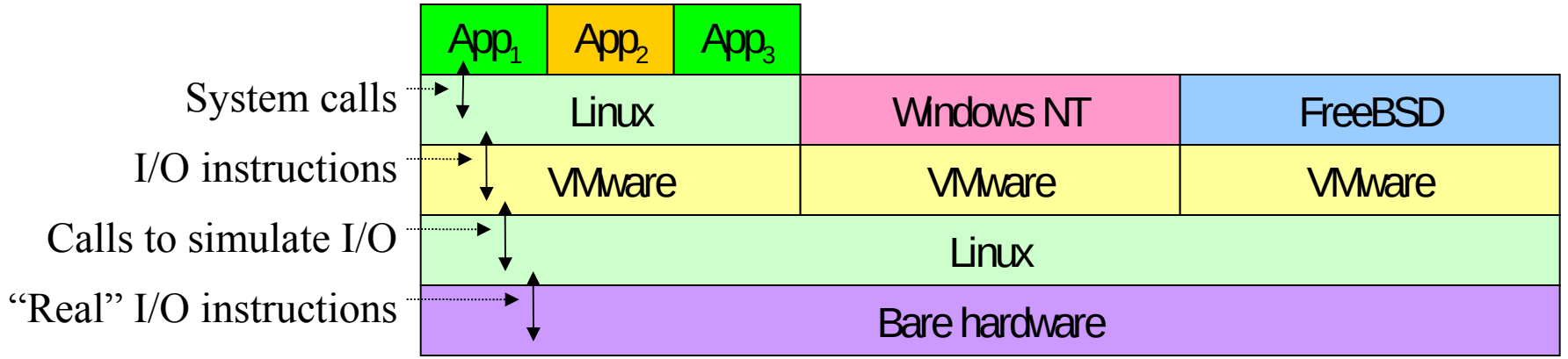
	FAT	FAT32	NTFS	ext2	ext3	Reiser	XF
Pääsu-õigused	-	-	+	+	+	+	+
ACL	-	-	+	+	+	(+)	+
Journal	-	-	+	-	+	+	+
Kvoodid	-	-	+	+	+	+	+
Krüpto	-	-	+	-	-	(+)	-
Kiirus						+	+

Virtual machines



Virtuaalkeskkondade kasutamine

- **Õppeotstarbel**
- **Testkeskkondadeks**
- **Tootearenduses ning kasutajatoes**
- **Server - hosting**
  - Kliendid saavad täiesti oma serveri
- **Serverite virtualiseerimine**
  - Riistvararessurss jaguneb mitme serveri vahel
  - Lihtsustub riistvara hooldus
  - Virtuaalservereid saab ümber tõsta teisele riistvarale
- **Teenuste eraldamiseks - igale teenusele oma server**
  - turvalisus



Virtualiseerimisvahendid

- **PC riistvara emuleerimine**
  - Vmware
    - Olemas nii Linux, kui Windows versioon
    - Eri versioonid töökoha ja serverirakendusteks
  - Virtual PC (Microsoft)
  - Parallels (MacOS)
- **Linuxipõhised virtualiseerimisvahendid**
  - Virtuozzo. OpenVZ
  - Vserver
  - XEN
  - UML
  - QEMU

Kokkuvõtteks

- **Operatsioonisüsteemi tuuma funktsioonid**
  - ressursside haldamine (mälu, protsessor, seadmed)
  - protsesside haldamine
  - võrguliides ja võrguprotokollid
  - turvalisuse garanteerimine
- **Operatsioonisüsteemi muud funktsioonid**
  - kasutajate andmebaas
  - tarkvarahaldus
  - kettaressursside haldus
  - monitooring